Algorithmic Modeling and Complexity (AMC)
Final exam/Tentamen (4 November 2003, 9.00-12.00)

This is an *open notes* exam: you may consult the scribe notes and your own notes from *this year's* AMC-course, but you may not consult other books or reference materials. Be complete in your answers. Every question counts for 1 point. Write your name and studentnumber on every sheet that you hand in. Good luck!

1. What is the m-TSP problem. Show that the m-TSP problem is NP-complete for every fixed value of $m \geq 1$.

2. Show that the metric, symmetric k-center problem is not $\in$ PTAS unless P=NP.

3. For integer constant δ, let MinDomSet$_\delta$ denote the Minimum Dominating Set problem restricted to networks with maximum node-degree $\leq \delta$. The decision version of MinDomSet$_\delta$ is known to be NP-complete for every fixed $\delta \geq 3$. Show that MinDomSet$_2$ $\in$ P. Is MinDomSet$_\delta$ $\in$ APX, for every $\delta \geq 3$? Why, or why not.

4. The Minimum Hitting Set problem is the following problem: given a universe $U = \{1, \ldots, n\}$ and a collection of subsets $\{S_1, \ldots, S_m\}$ of U, determine a minimum cardinality subset $C \subseteq U$ such that $C \cap S_i \neq \emptyset$ for every $1 \leq i \leq m$. The problem arises when one wants to determine fair representations from subsets. Give a 0-1 LP model for the Minimum Hitting Set problem. Show that the Minimum Dominating Set problem (without weights) can be reduced to the Minimum Hitting Set problem. What does this imply for the complexity of the Minimum Hitting Set problem?

5. Consider the maximum cardinality bipartite matching problem. Show that in any given instance, an arbitrary maximal matching always contains a number of matchings that is at least half the number of matchings in a maximum matching.

6. Suppose the Minimum Makespan Scheduling problem for identical machines can be solved in polynomial time. Of course this would imply that P=NP, but assume that this does not lead to a practical transformation of any NP-algorithm to polynomial-time. Show that the Minimum Bin Packing problem can be solved in polynomial-time with the help of the Minimum Makespan Scheduling solver. How many calls of the solver are needed?

7. Determining minimum dominating sets is NP-hard. One may try to avoid the problem by looking for (large) collections of disjoint dominating sets. Let $G = (V, E)$ be an arbitrary network. Define $DN(G)$, the so-called *domatic number* of G, as the largest k such that V can be partitioned into k disjoint subsets $V_1, \ldots, V_k$ which all are dominating sets of G. The domatic number is well-defined and the following problem is known to be NP-complete: given a network G, is $DN(G) \geq 4$.

Can there exist a polynomial-time algorithm A that partitions every network G into $d_A(G)$ disjoint dominating sets such that $d_A(G) \geq c \cdot DN(G)$, for certain $c > \frac{4}{5}$? Under what condition(s) would it, or would it not be possible.

8. Several fault-tolerant distributed algorithms that were presented were proved correct provided we restrict ourselves to fair executions.

(a) Give a definition for each of the following three concepts: (i) fair t-initially dead execution, (ii) fair t-crash execution, (iii) fair t-Byzantine execution.

(b) In lecture 17 we gave a Byzantine broadcasting algorithm A for synchronous networks. Theorem 17.3 states the correctness of A but does not require executions to be fair. Discuss the question whether A is correct for executions that are not fair.

(c) Can you give a general definition of fair executions in the context of fault-tolerant distributed algorithms? Your definition should not contradict the three definitions you gave for question (a).

9. In lecture 16 a distributed consensus algorithm for N processes was presented in which at most $t < N/2$ processes are initially dead. It was shown to be correct for fair executions. Now suppose that one of the initially dead processes (say process P_1) unexpectedly comes alive after some time and starts to execute.

Discuss the question whether in this situation the algorithm is still correct for fair executions. Does a possible correctness depend on the moment that P_1 comes alive? That is, does it make a difference in the answer of the question whether P_1 comes alive when the other processes have hardly started or have nearly determined their output?

10. What role can algorithmic modelling play in software system design. Give and explain at least three issues for which algorithmic modelling is crucial. What limitations does algorithmic modelling have?